

BELEUCHTUNG UND RENDERING I

BELEGARBEIT

Dokumentation von

Daniel Kegel, Eric Schmidtgen und Linda Garten

Betreuender Hochschullehrer

Prof. Dr. Marco Block-Berlitz

Eingereicht am: 22. Januar 2020

INHALTSVERZEICHNIS

1	Einleitung und Aufgabenstellung	3
2	Vorstellung der Lösung	4
2.1	Idee	4
2.2	Unity	4
2.3	Unity - Asset Store	4
2.4	Erster Überblick	5
2.5	Klassen	5
2.5.1	First Person Controller	5
2.5.2	Flashlight	5
2.5.3	Reset	6
2.5.4	MouseLook	6
2.5.5	PlayerFollow	6
2.5.6	Tastaturbelegung	7
2.6	Welt - DesertSetHouse	7
2.7	Probleme beim Import	7
2.8	Ergebnis	7
3	Abbildungsverzeichnis	8

1 EINLEITUNG UND AUFGABENSTELLUNG

In Gruppen von bis zu drei Personen dürfen kleine Projektideen von der Konzeption bis zur Umsetzung durchgeführt werden. Die Inhalte müssen allerdings mit den Dozenten vorab abgesprochen werden. Grundvoraussetzung für die Projektideen sind Innovation und Interaktion. Es dürfen auch Game-Engines verwendet werden. Bei der Bewertung wird der Eigenanteil entsprechend berücksichtigt. Teil 2 fließt mit mindestens 50 kompensieren. In jedem Fall ist die Rücksprache mit den Dozenten erforderlich.

Das Thema für die Projekte lautet in diesem Semester: SCHATTEN

Die Abgabe der Arbeit erfolgt im letzten Praktikum des Wintersemesters 2019/20.

2 VORSTELLUNG DER LÖSUNG

2.1 IDEE

Unsere Idee ist es, dass wir in UNITY eine Taschenlampe als First Person durch eine Welt bewegen können. Wir sind dabei die Taschenlampe und können uns durch eine Welt mit einem bewegen. Unity bietet den Vorteil, dass einfache Objekte wie Lichtquellen oder grafische Primitive (Ebenen, Würfel, Kugeln) einfach im Editor erzeugt werden können. Somit sparen wir uns die Berechnung des Lichtes beziehungsweise des Schattens und wir erzielen schnell Erfolge. Mit selbst erstellten Skripten können wir Mausbewegungen oder Tastatureingaben übertragen.

2.2 UNITY

1. Versuch

Das Projekt wurde mit der Unity Version UNITY 2018.4.14 gebaut, welche am 11.12.2019 veröffentlicht wurde. Später im Projekt wird uns das zum Verhängnis, weil wir das Projekt einmal mit einer aktuelleren Version geöffnet haben und uns Unity damit Projektdateien zerstört hat. Somit war es nicht mehr möglich, dass Projekt mit irgendeiner Unity Version zu öffnen.¹

Durch unsere Probleme mit den verschiedenen Unity-Versionen haben wir nun versucht auf einen älteren, hoffentlich noch funktionierenden, Stand/Commit zurückzugehen.² Auch bei diesem Vorgehen ist Unity immer ohne Fehlermeldung abgestürzt.³

Am 22.01.2020 haben wir ein neues Repository angelegt.⁴

2. Versuch

Da wir unser Projekt nicht mehr öffnen konnten, haben wir uns dazu entschieden es noch einmal neu aufzubauen - gleich in der neusten Unity-Version 2019.2.18. Zudem legen wir ein neues Repository an um uns abzusichern.

GITHUB

Da mehrere Personen an einem Projekt arbeiten und wir immer einen aktuellen Stand haben wollten, haben wir uns für den Einsatz von GitHub und einem Repository entschieden.

2.3 UNITY - ASSET STORE

Damit wir nicht viel Zeit beim modellieren aufwenden wollen, werden wir auf bereits fertige ASSETS zurückgreifen. Für die Taschenlampe verwenden wir das freie Asset von RRFREELANCE⁵.

¹Dieses Ereignis beobachteten wird bereits ab dem 15.01.2020.

²Das erste Mal wurde dieses Vorgehen am 21.01.2020 versucht.

³Link zum alte Repository: https://github.com/DanielKegel/movement_unity

⁴Link zum neuen Repository: <https://github.com/DanielKegel/Beleg-V2>

⁵RRFreelance, "Flashlight", im Unity Asset Store, <https://assetstore.unity.com/packages/3d/props/electronics/flashlight-18972>, abgerufen am: 14.01.2020

2.4 ERSTER ÜBERBLICK

Nach ein paar Stunden im Probieren mit Unity (Rigidbody, Gravity, Animator, etc.) laden wir unser Asset (siehe Abb. 1). Wir probieren erst einmal die schon fertigen Komponenten des Assetes und stellen schnell fest, dass wir mit der Physik nicht zufrieden sind. Wir bemerken, dass die Physik der Kamera in Kombination mit dem Licht nicht stimmig ist (siehe Abb. 2). Dies werden wir mit weiteren Klassen anpassen, da eine Taschenlampe und Kamera nur mit einem anderen Körper gut harmonisieren.

2.5 KLASSEN

2.5.1 FIRST PERSON CONTROLLER

Der First Person Controller (FPC) ist eine Kombination aus einer Kamera und einem Lautsprecher verpackt in einem Objekt - in diesem Fall ein TICTAC. In dieser Gruppe ist auch die Taschenlampe untergeordnet. Der Controller bietet uns eine vielfältige Auswahl an Tasten- und Mausbelegungen und Sounds. So hören wir zum Beispiel einen Sound beim laufen, springen und wieder landen. Alle Sounds sind individuell anpassbar.

2.5.2 FLASHLIGHT

Die Taschenlampe bekommt von Unity das Attribut LICHT zugewiesen mit einer Intensität von 2 und einem SPOT ANGLE von 50.

Um die Physik der Taschenlampe anzupassen, müssen wir ein Script anlegen und etwas C# programmieren.

Am Anfang legen wir ein paar Variablen an und initialisieren diese direkt mit Werten. Wir benötigen diese später für unsere Physik.

```
1 public class FlashLight : MonoBehaviour {
2     public float horizontalSpeed = 2f;
3     public float verticalSpeed = 2f;
4     float mainSpeed = 100.0f; //regular speed
5     float shiftAdd = 250.0f; //multiplied by how long shift is held.
6     Basically running
7     float maxShift = 1000.0f; //Maximum speed when holdin gshift
8     float camSens = 0.25f; //How sensitive it with mouse
9     private Vector3 lastMouse = new Vector3(255, 255, 255); //kind of
10    in the middle of the screen, rather than at the top (play)
11    private float totalRun = 1.0f;
12    ...
13 }
```

In der UPDATE() - Funktion lesen wir die X und Y-Werte der Mausposition aus und geben diese an die Transformation des Elementes weiter.

```
1 void Update() {
2     float h = horizontalSpeed * Input.GetAxis("Mouse X");
3     float v = verticalSpeed * Input.GetAxis("Mouse Y");
4     this.transform.Rotate(v, 0, h);
5     ...
6 }
```

Damit wir uns nun noch in der Welt mit unserer Taschenlampe bewegen können, implementieren wir uns noch ein paar Tastenkombinationen. Dafür arbeiten wir mit Vector3D.

```
1 private Vector3 GetBaseInput() {
2     Vector3 p_Velocity = new Vector3();
3     if (Input.GetKey(KeyCode.W)) {
4         p_Velocity += new Vector3(0,0.03f,0);
5     }
6     if (Input.GetKey(KeyCode.S)) {
7         p_Velocity += new Vector3(0, -0.03f,0);
8     }
9     if (Input.GetKey(KeyCode.A)) {
10        p_Velocity += new Vector3(0.03f, 0, 0);
11    }
12    if (Input.GetKey(KeyCode.D)) {
13        p_Velocity += new Vector3(-0.03f, 0, 0);
14    }
15    return p_Velocity;
16 }
```

2.5.3 RESET

Da unsere Welt, insbesondere unser Haus, etwas fehlerbehaftet ist und wir oft mit unserem Spieler in Wänden stecken bleiben oder wir GLITCHEN, haben wir uns dafür entschieden einen Reset-Button einzufügen. Beim Klicken der Taste R wird der Spieler an die Position (0,1,0) zurückgesetzt. Die haben wir mit einem 3D-Vektor abgebildet:

```
1 if (Input.GetKeyDown(KeyCode.R)) {
2     transform.position = new Vector3(0, 1, 0);
3 }
```

2.5.4 MOUSELOOK

Die Klasse MOUSELOOK rotiert und transformiert auf Basis unseres Maus-Deltas. Ein Minimum und Maximum begrenzt unsere Rotation. Das erstellte Script wird der Kamera zugewiesen, welche ein Kind des TIC-TAC's ist.

2.5.5 PLAYERFOLLOW

Damit der Spieler sich so natürlich wie möglich bewegt, brauchen wir noch einen gewissen SMOOTH-Faktor. Dieser beeinflusst die Bewegung deutlich. Die Richtung des zurückgegeben Vektors wird durch den Winkel interpoliert und seine Größe wird zwischen den Größen von aktueller Position und neuer Position interpoliert.

```
1 void Update () {
2     Vector3 newPos = PlayerTransform.position + _cameraOffset;
3     transform.position = Vector3.Slerp(transform.position, newPos,
4         SmoothFactor);
5     if (LookAtPlayer)
6         transform.LookAt(PlayerTransform);
7 }
```

2.5.6 TASTATURBELEGUNG

W	Bewegung nach vorn
A	Bewegung nach links
S	Bewegung nach hinten
D	Bewegung nach rechts
R	Zurücksetzen
F	Licht aus bzw. anschalten
G	Lichtfarbe wechseln
LEFT SHIFT	Sprint
SPACE	Springen

2.6 WELT - DESERTSETHOUSE

Damit wir mit unserer Taschenlampe nicht einfach auf einer einfachen und langweiligen Oberfläche laufen, werden wir ein altes Wüstengebäude nutzen. Auch dies haben wir im Unity AssetStore gefunden⁶. Damit es für unser Projekt passt, mussten wir noch einige Anpassungen in Maya vornehmen. So haben wir die Türen geöffnet, damit wir sauber mit der Taschenlampe durchlaufen können (siehe Abb. 3). Zudem wurden noch einige Wände vervollständigt (siehe Abb. 4).

2.7 PROBLEME BEIM IMPORT

Beim Import des Maya-Files in Unity gab es Probleme mit dem Schatten und den Texturen. So wurden die Wände im Haus nicht gezeichnet beziehungsweise ausgeleuchtet. Nach einer kurzen Recherche haben wir den Fehler gefunden und konnten mit Hilfe von DOUBLE-SIDED-SHADERS das Problem fixen⁷.

2.8 ERGEBNIS

Mit dem abschließenden Ergebnis sind wir sehr zufrieden. Wir alle haben das erste mal mit Unity gearbeitet und waren überrascht, wie schnell man Fortschritte macht und zu einem Ergebnis kommt (siehe Abb. 5). Die Möglichkeiten schienen schier unbegrenzt.

Wir haben einen Spieler, den man definierten Tasten bewegen kann. Umschauen kann man sich mit der Maus. So ist es möglich durch das Haus zu navigieren (siehe Abb. 6 und 7).

⁶Crazy Cool, "Desert Set House", im Unity Asset Store, <https://assetstore.unity.com/packages/3d/environments/historic/desert-set-house-138105>, abgerufen am: 14.01.2020

⁷Ciconia Studio, "Double Sided Shaders", im Unity Asset Store, <https://assetstore.unity.com/packages/vfx/shaders/double-sided-shaders-23087>, abgerufen am: 14.01.2020

3 ABBILDUNGSVERZEICHNIS

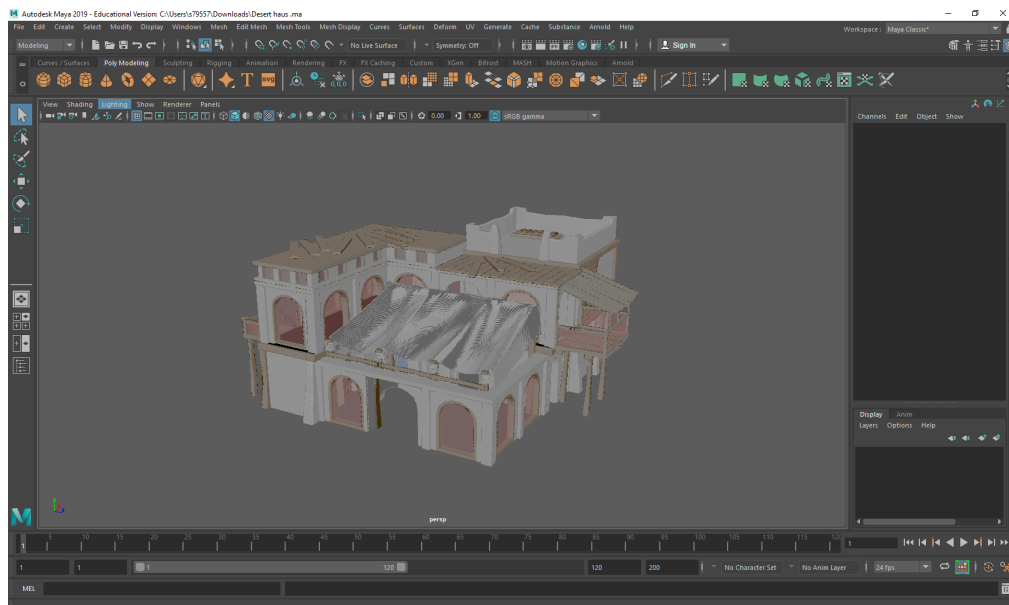


Abbildung 3: Autodesk Maya - Sicht von vorn

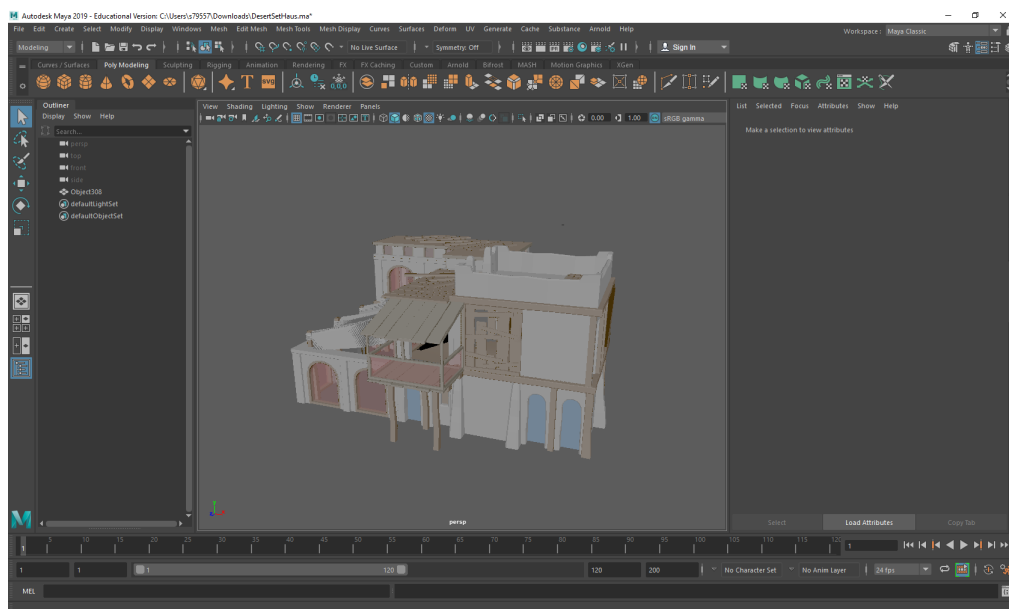


Abbildung 4: Autodesk Maya - Sicht von der Seite

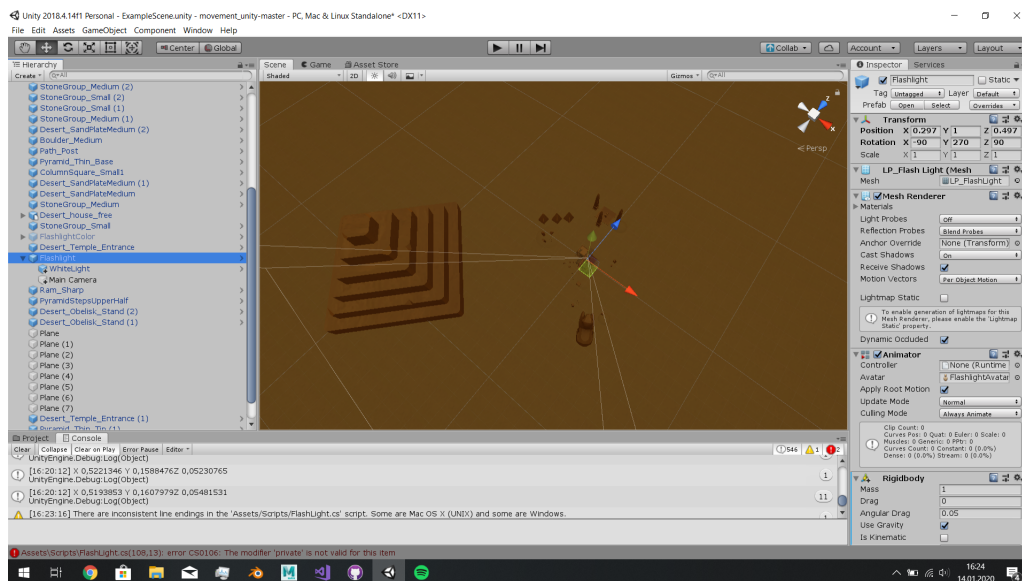


Abbildung 5: Unity UI - Überblick über die Welt; links der Outliner des Projektes

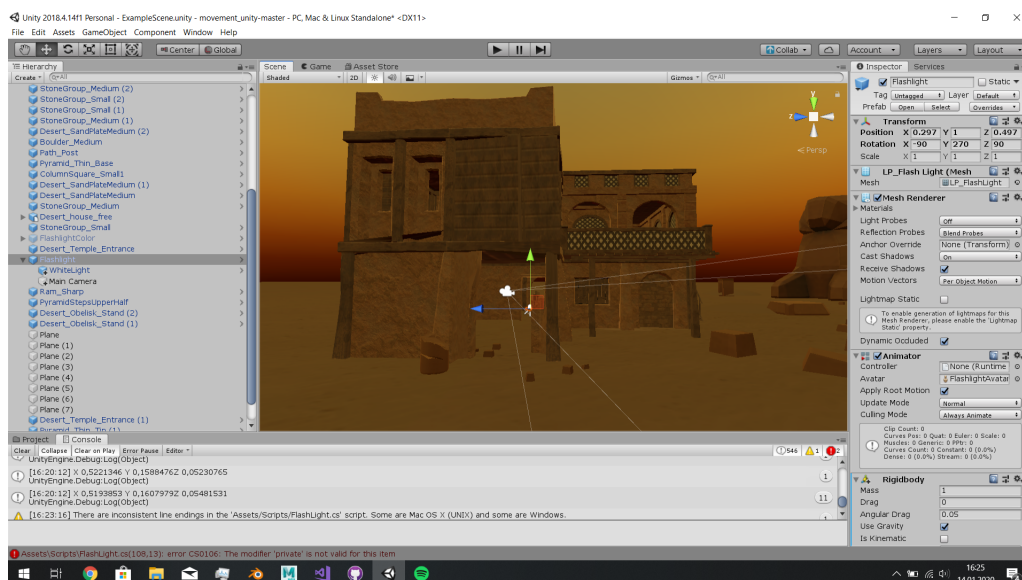


Abbildung 6: Unity UI - Blick auf das Desert House

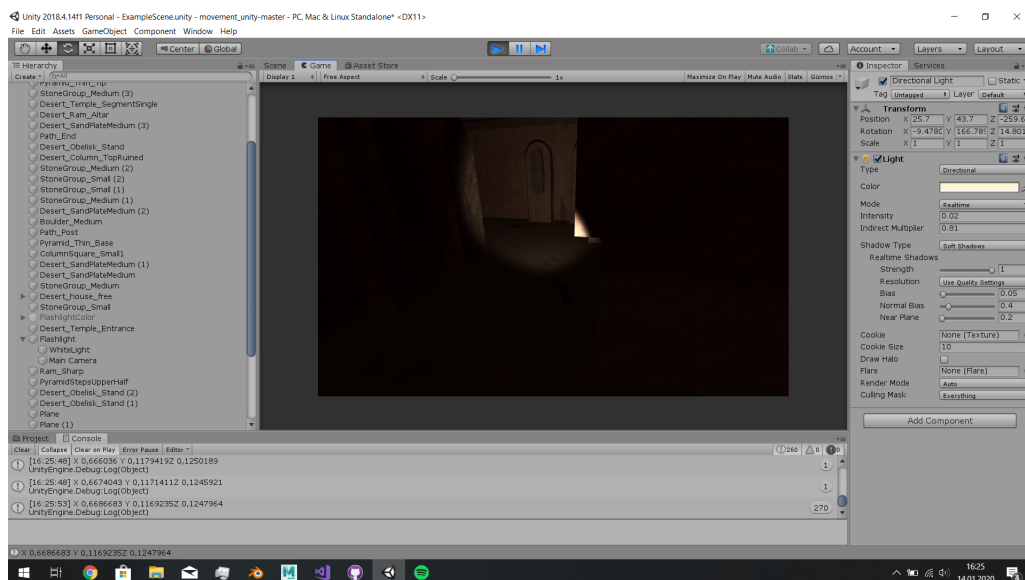


Abbildung 7: Unity UI - Projekt im Run Modus